JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11)Publication number: 10124331
(43)Date of publication of application: 15.05.1998

(51)Int.Cl. G06F 9/46

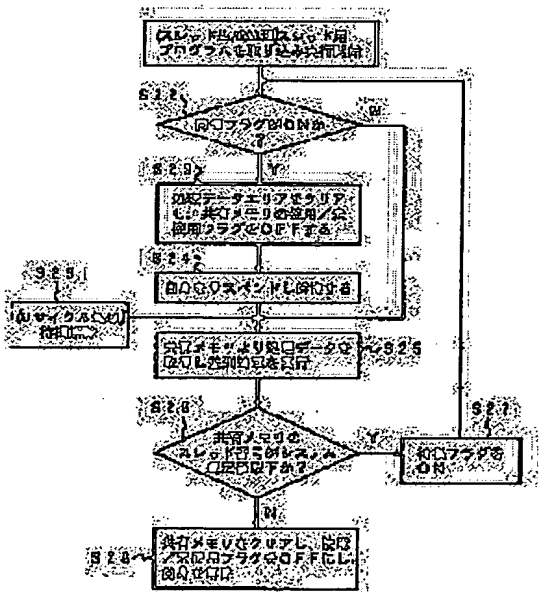
(21)Application number: 08275885
(22)Date of filing: 18.10.1996

(71)Applicant: NEC ENG LTD
(72)Inventor: YOSHIMOTO IKUKO

(54) DATA PROCESSING METHOD

(57)Abstract:

PROBLEM TO BE SOLVED: To prevent a system from the lack of resources and the reduction of performance due to the increment of threads by reducing the number of stand-by threads up to the number of operating threads when the number of stand-by threads after parallel processing is larger than the number of threads required for system operation.
SOLUTION: A thread program is entered. While referring to a common memory, whether a stand-by flag is ON or not is checked (S22). Just after the generation of a thread, a stand-by flag corresponding to the thread is OFF. Then processing data are acquired from the common memory and processing for executing parallel processing is executed (S25). Then whether a thread number in the common memory is smaller than a regulated value indicating the number of threads at the time of normal system operation or not is checked (S26). When the thread number is larger than the regulated value, a use/unused flag, a stand-by flag and processing data in the common memory are cleared (S28).



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平10－124331

(43)公開日　平成10年(1998)5月15日

(51)Int.Cl.[6]	識別記号	FI	
G06F 9/46	340	G06F 9/46	340B
			340D

審査請求　未請求　請求項の数2　ＯＬ　（全 10 頁）

(21)出願番号　特願平8－275885

(22)出願日　平成８年(1996)10月18日

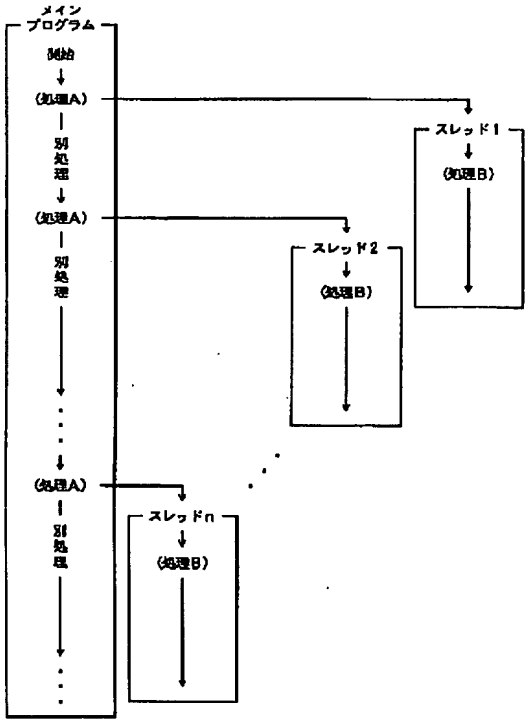
(71)出願人　000232047
日本電気エンジニアリング株式会社
東京都港区芝浦三丁目18番21号

(72)発明者　吉本　郁子
東京都港区芝浦三丁目18番21号　日本電気エンジニアリング株式会社内

(74)代理人　弁理士　鈴木　正剛

(54)【発明の名称】　データ処理方法

(57)【要約】
【課題】　スレッドの増加によるシステムの資源不足や性能低下を防止できるデータ処理方法を提供する。
【解決手段】　データ処理システム上で複数のスレッドにより並行処理を行うデータ処理方法である。並列処理の実行後に、データ処理システムのシステム運用に必要となる規定値の個数のスレッドを待機状態とし、この規定値の個数を越えたスレッドを並列処理の実行後に削除する。



1

【特許請求の範囲】

【請求項１】　所定のデータ処理システム上で複数のスレッドにより並行処理を行うデータ処理方法において、並列処理の実行後に、前記データ処理システムのシステム運用に必要となる規定値の個数のスレッドを並行処理が再開可能な待機状態とし、前記規定値の個数を越えたスレッドを並列処理の実行後に削除することを特徴とするデータ処理方法。

【請求項２】　並列処理に使用される各スレッドに対して所定のスレッド番号を順次設定するとともに並行処理の実行直後に各スレッドのスレッド番号と前記規定値とを比較し、スレッド番号が前記規定値より大きいスレッドは削除し、スレッド番号が前記規定値以下であるスレッドは前記待機状態とすることを特徴とする請求項１記載のデータ処理方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、データ処理方法に関し、特に、データ処理システムにおいてスレッドの生成や削除などを管理する方法に関する。

【０００２】

【従来の技術】例えばコンピュータを用いたデータ処理システムにおいて、メインプログラムの実行中に、並行処理、例えば、メインプログラムの処理と並行して特定の関数を実行する場合がある。上記並行処理は、例えばスレッドを用いて行われる。このスレッドを用いる場合、一度生成したスレッドを削除することなく再利用することで、スレッドの生成時間を短縮・削減し、システムにおける処理の高速化を図ることも試みられている。このような技術については、例えば特開平４－２３１３７号や特開平６－３１８１９１号公報などに開示されている。このようなスレッドを用いた従来のデータ処理の一例を、図６～図８を用いて以下に説明する。

【０００３】まず、図６において、リサイクル可能なスレッドの個数をチェックする（ステップＳ３１）。このような個数チェックは、図８のリサイクルスレッドカウントＧのカウント値から判別できる。そして、この個数が０の場合には、リサイクル可能なスレッドが存在しないことから、ステップＳ３６以降の処理に移行し、スレッドの生成処理を行う。一方、上記の個数が０でなく、リサイクル可能なスレッドが存在する場合は、スレッド管理テーブルの先頭から各スレッドの使用／未使用フラグＢを順次チェックし、これらの使用／未使用フラグの状態をチェックする（ステップＳ３２）。

【０００４】そして、使用／未使用フラグがオフで未使用のスレッドがある場合は、そのスレッドの使用／未使用フラグをオン、つまり使用中にし（ステップＳ３３）、また、図８の処理データＤに処理に必要なデータをセット（ステップＳ３４）する。さらに、図８のスレッド番号（ハンドル値）Ａに基づいてスレッドの再開処理を行う（ステップＳ３５）。

2

【０００５】一方、使用／未使用フラグがオフで未使用のスレッドが存在しない場合は、テーブルの先頭から上記のリサイクル可能なスレッド個数分だけ飛ばした位置にある、スレッド管理テーブル上の空きエリアを確保し、この空きエリアにおける使用／未使用フラグを使用中とする（ステップＳ３６）。また、この空きエリアに、スレッド番号として、スレッド管理テーブルの先頭から何番目のエリアであったかをセットする（ステップＳ３７）。そして、上記のようにスレッドが生成され、リサイクル可能なスレッド数が増えたことから、図８のリサイクルスレッドカウントＧを＋１する（ステップＳ３８）。

【０００６】次に、上記でセットしたエリアをスレッドとメインプログラムとの共用メモリに指定するため、スレッド生成時において指定する共用メモリアドレスとして、データをセットしたスレッド管理テーブル内のエリアのアドレスを、図８のＥのアドレスに割り当てる（ステップＳ３９）。これにより、スレッドの生成処理が完了し、その後、ステップＳ３４とステップＳ３５の処理を行ってスレッドの実行の指示をする。

【０００７】ここで、共用メモリは、スレッドが参照して動作するためのものである。そして、スレッドを必要とする場合には、管理テーブルから未使用のスレッドを検索し、未使用スレッドに該当する管理テーブル内のエリアの使用／未使用フラグを使用中とし、スレッドの動作に必要なデータを所定のエリア内にセットし、その後、親スレッドが動作のトリガとして、そのスレッドのリジウムを行い、処理動作を指示する。また、スレッドは、割り当てられた親スレッドとの共用メモリエリアから必要なデータを取得し、動作を開始する。さらに、この共用メモリを書き替えることで、親スレッドはスレッドに対して所定の動作を指示し実行させることができる。

【０００８】また、図７は、スレッドにおける並行処理の手順を示したもので、スレッドは、スレッド用プログラムを取り込んで並行処理の実行を開始する。まず、スレッドは、その生成後すぐに、それ自身をサスペンドして待機状態とする（ステップＳ４２）。なお、サスペンドとは、スレッドを中断して再開可能な待機状態にすることをいう。そして、スレッドは、上記のような再開の指示があった場合、共用メモリＦから処理データを取得し、並行処理を実行する（ステップＳ４３）。さらに、並行処理実行後は、処理データエリアをクリアし、使用／未使用フラグＢを未使用にセットし、再び待機状態になる（ステップＳ４２）。

【０００９】以上のようにして、全てのスレッドがリサイクル可能なスレッドとなり、また、使用されないときには常に待機状態となる。さらに、スレッドの削除は、システムのシャットダウン時までは行われない。

【００１０】

【発明が解決しようとする課題】ところで、上記のようなデータ処理システムにおいてシステムに負担がかかった場合には、当然のことながら、スレッドの個数が通常のシステム運用時の個数よりも増大する。そして、上記従来の方法の場合、スレッドの個数は、例えば、図９のように、通常運用時の個数よりも増加する。

【００１１】ところが、この従来は、スレッドの個数が増大した後に通常の運用に戻った場合、図９のようにスレッドの個数が増大したままとなる。このため、使用されない複数のスレッドが待機状態となって、システムの資源を無駄に使用し続けるという問題があった。

【００１２】また、スレッドは、待機状態であっても、非常に少ない値ではあるがＣＰＵ時間を消費する。よって、上記従来の手法では、システムに負荷がかかった後に通常の運用に戻った際、待機状態のスレッドによるＣＰＵ時間の消費量が増加してしまう。このため、通常の運用に戻った際の運転時におけるシステムの処理性能の低下を招くという問題もあった。

【００１３】本発明は、上記のようなスレッドの増加によるシステムの資源不足や性能低下を防止できる、改良されたデータ処理方法を提供することを課題としている。

【００１４】

【課題を解決するための手段】本発明のデータ処理方法は、所定のデータ処理システム上で複数のスレッドにより並行処理を行うデータ処理方法において、並列処理の実行後に、前記データ処理システムのシステム運用に必要となる規定値の個数のスレッドを並行処理が再開可能な待機状態とし、前記規定値の個数を越えたスレッドを並列処理の実行後に削除することを特徴とする。

【００１５】ここで、システム運用に必要となる規定値の個数とは、システム上における通常の運用において使用する任意の個数ないし規定値を意味する。このような規定値は、システムに応じて変化するものであり、システムに応じた任意の値とすれば良い。

【００１６】つまり、本発明では、並行処理後における待機スレッドの数がシステム運用に必要なスレッド（運用スレッド）の数よりも多い場合には、待機スレッドの数を運用スレッドの数まで削減するようにしたものである。

【００１７】上記のような並行処理実行後におけるスレッドの削除を行うため、例えば、並列処理に使用される各スレッドに対して所定のスレッド番号を順次設定するとともに並行処理の実行直後に各スレッドのスレッド番号と前記規定値とを比較し、スレッド番号が前記規定値より大きいスレッドは削除し、スレッド番号が前記規定値以下であるスレッドは前記待機状態とする、手順が用いられる。

【００１８】

【発明の実施の形態】以下に、本発明のデータ処理方法の実施の形態を説明する。図１は、本発明が適用されるデータ処理システムにおける処理の概要を示したものである。そして、このメインプログラムの実行中において、複数のスレッドにより並行処理を行う必要が生じた場合には、メインプログラムにおいては以下に説明する処理Ａが、また各スレッド（スレッド１～スレッドｎ）においては以下に説明する処理Ｂがそれぞれ行われる。

【００１９】まず、データ処理システムのメインプログラムにおける処理Ａについて、つまり、メインプログラムにおけるスレッド管理処理について、図２を参照して説明する。この処理は、次の（１）～（４）のように行われる。

（１）　並行処理が必要となった時は、リサイクルスレッドカウントが０であるかをチェックするとともに、リサイクルスレッドが存在するかどうかをチェックする（ステップＳ１）。

【００２０】なお、リサイクルスレッドカウンタは、例えば、図４に符号Ｇとして例示したように、データ処理システムの所定のスレッド管理テーブルに設定されるもので、スレッド管理テーブル内におけるリサイクル可能なスレッドの数を示すものである。また、このスレッド管理テーブルには、このテーブルが管理しているスレッドの数に対応するだけの共用メモリＦがあり、各共用メモリＦは、スレッド番号（ハンドル値）Ａ、使用／未使用フラグＢ、待機フラグＣ、処理データＤがそれぞれ設定されている。

【００２１】ここで、スレッド番号Ａは、スレッド管理テーブルにおいて管理されるスレッドに対して順次設定される番号を示したものである。また、使用／未使用フラグＢは、各スレッドの使用状態を示すためのものである。さらに、待機フラグＣは、そのスレッドが生成直後ではなくてリサイクル可能なスレッドであることを示すためのものである。さらに、処理データＤは、メインプログラムから並行処理に必要なデータをスレッドへ提供するためのものである。

【００２２】そして、リサイクルカウンタが０であった場合、つまり、リサイクルスレッドが存在しない場合には、以下の（４）の処理に進み、スレッドの生成処理を行う。また、リサイクルカウンタが０でない場合には、以下の（２）の処理に進み、利用可能なリサイクルスレッドが存在するかどうかのチェックを行う。

【００２３】（２）　リサイクルスレッドカウントに対応する数のスレッド管理テーブル内の使用／未使用フラグＢをチェックする（ステップＳ２）。そして、未使用のリサイクルスレッドがない場合は、下記（４）のスレッド生成処理に進む。また、未使用のリサイクルスレッドがある場合は、以下の（３）のリサイクルスレッドを利用した処理へ進む。

【００２４】（３）　リサイクルスレッドの利用は、ま

ず、使用しようとしているスレッドの共用メモリに指定される管理テーブルの待機フラグＣをオンにする（ステップＳ３）。次に、使用／未使用フラグＢをオンにして使用中とし（ステップＳ４）、並行処理に必要なデータを処理データＤにセットし（ステップＳ５）、スレッドを再開させる（ステップＳ６）。すると、スレッドは、ステップＳ２９より待機状態から解かれて再開され、図３に示したリサイクル処理（Ｓ２９）が行われる。

【００２５】（４）スレッドの生成処理は次のように行われる。まず、使用しようとしているスレッドの共用メモリＦの待機フラグＣをオフにする（ステップＳ７）。次に、その共用メモリＦにおける使用／未使用フラグＢを使用中とする（ステップＳ８）。さらに、スレッド管理テーブルの共有メモリＦのスレッド番号Ａをスレッド番号Ａを設定し、またた並行処理に必要なデータを処理データＤにセットする（ステップＳ９）。次に、今後このスレッドをリサイクルスレッドとして利用するかどうかの判断として、スレッド番号Ａが、データ処理システムにおいて通常使用するスレッド数（規定値）より大きいかをチェックし（ステップＳ１０）、小さい場合はリサイクルスレッドとして利用することから、スレッド管理テーブルのリサイクルスレッドカウントＧを＋１する（ステップＳ１１）。そして、スレッド管理テーブルの上記で設定およびセットを行ったエリアの共用メモリＦをメインプログラムとスレッドとの共用メモリに指定して、スレッドの生成を行う（ステップＳ１２）。なお、このようにして生成されたスレッドは、図３のステップＳ２１から処理を開始する。

【００２６】次に、各スレッド内での処理を、図３により説明する。この処理は、スレッドの生成直後における処理と、スレッドのリサイクル時における処理の２つに分かれる。

【００２７】まず、スレッドの生成直後におけるスレッド内での処理は次の通りである。最初に、スレッド生成処理として、スレッド用プログラムが取り込まれ、スレッド生成処理が開始される（ステップＳ２１）。次いで、共用メモリＦを参照して、待機フラグＣがオンであるかどうかをチェックする（ステップＳ２２）。ここで、スレッド生成直後においてはそのスレッドに対応する待機フラグＣはオフになっている。そして、次に、共用メモリＦから処理データＤを取得し、並行処理を実行する処理が行われる（ステップＳ２５）。

【００２８】次いで、共用メモリＦのスレッド番号Ａが通常のシステム運用時におけるスレッド数の規定値より小さいかどうかをチェックする（ステップＳ２６）。そして、スレッド番号Ａが規定値よりも大きい場合は、その共用メモリＦにおける使用／未使用フラグＢ、待機フラグＣ、並びに処理データＤがそれぞれクリアされ、終了する（ステップＳ２８）。なお、使用／未使用フラグＢをクリアすることは、この共用メモリＦを未使用にすることを意味する。また、処理データＤをクリアすることは、そのスレッドを削除することを意味する。

【００２９】また、スレッド番号が規定値よりも小さいか、または同じであった場合は、その共用メモリＦにおける待機フラグＣをオンにして（ステップＳ２７）、ステップＳ２２に処理が移行する。そして、ステップＳ２２では、待機フラグＣがオンであるので、リサイクル処理であるステップＳ２３、つまり、処理データＤのクリアと使用／未使用フラグＢを未使用にセットする処理が行われる。次いで、ステップＳ２４において、そのスレッドはそれ自身をサスペンドして待機状態となる。

【００３０】次に、スレッドのリサイクル時における処理を説明する。この処理は、次の（１１）～（１２）のように行われる。

【００３１】（１１）　すなわち、リサイクル時においては、スレッドはステップＳ２９のサスペンドして待機している状態から始まる。そして、メインプログラムから待機が解除されるので、共用メモリＦから処理データＤを取得し、並行処理を実行する（ステップＳ２５）。

【００３２】（１２）　次いで、共用メモリのスレッド番号Ａが通常のシステム運用時におけるスレッド数の規定値より小さいかどうかをチェックする（ステップＳ２６）。そして、スレッド番号Ａがこの規定値よりも以下であるので、共用メモリＦの待機フラグＣをオンにして（ステップＳ２７）、ステップＳ２２に処理を移す。そして、ステップＳ２２において、待機フラグＣがオンであるので、リサイクル処理であるステップＳ２３とステップＳ２４が行われる。

【００３３】以上のように構成されるこの実施の形態のデータ処理方法では、システムに負荷が発生して一時的にスレッド個数が図５のｌのように増大しても、並列処理の終了後は、図５でｌｌのように通常のシステム運用に必要なスレッド以外は削除され、これら必要なスレッドのみ待機状態となりリサイクル可能なスレッドとなる。

【００３４】このため、従来のように必要のないスレッドが待機状態でいるために使用されるＣＰＵ時間の消費が削減されて、負荷発生後におけるシステムの性能低下が防止できる。また、無駄なスレッドが生成され存在し続けることで使用されるシステム資源の無駄使いがなくなる、システムの有効な資源確保が行える。さらに、システムで正常時に必要なスレッドの数のスレッドを待機状態でリサイクルする構成であるので、その分のスレッドの生成や削除にかかる時間が削減されてシステムの性能向上が図れる。

【００３５】

【発明の効果】以上の説明から明らかなように、本発明によれば、スレッドの増加によるシステムの資源不足や性能低下が抑制される効果がある。

【図面の簡単な説明】

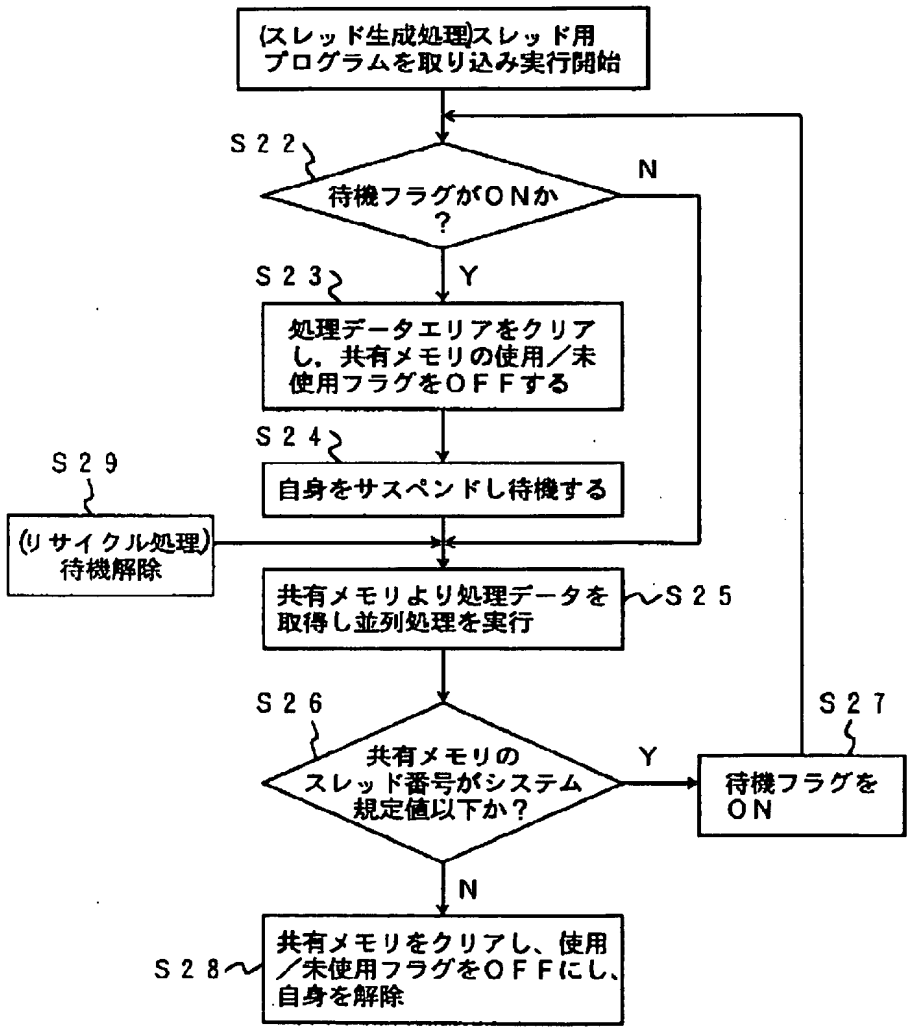
7

【図１】本発明が適用されるデータ処理システムにおける処理の概要を示した説明図。
【図２】実施の形態のデータ処理システムにおけるスレッド管理処理を示したフローチャート。
【図３】実施の形態のデータ処理システムにおける各スレッド内での処理を示したフローチャート。
【図４】実施の形態のデータ処理システムにおけるスレッド管理テーブルの説明図。
【図５】実施の形態のデータ処理システムにおけるスレッド数の推移を示した説明図。
【図６】従来のデータ処理システムにおける処理を説明したフローチャート。
【図７】従来のデータ処理システムにおける処理を説明

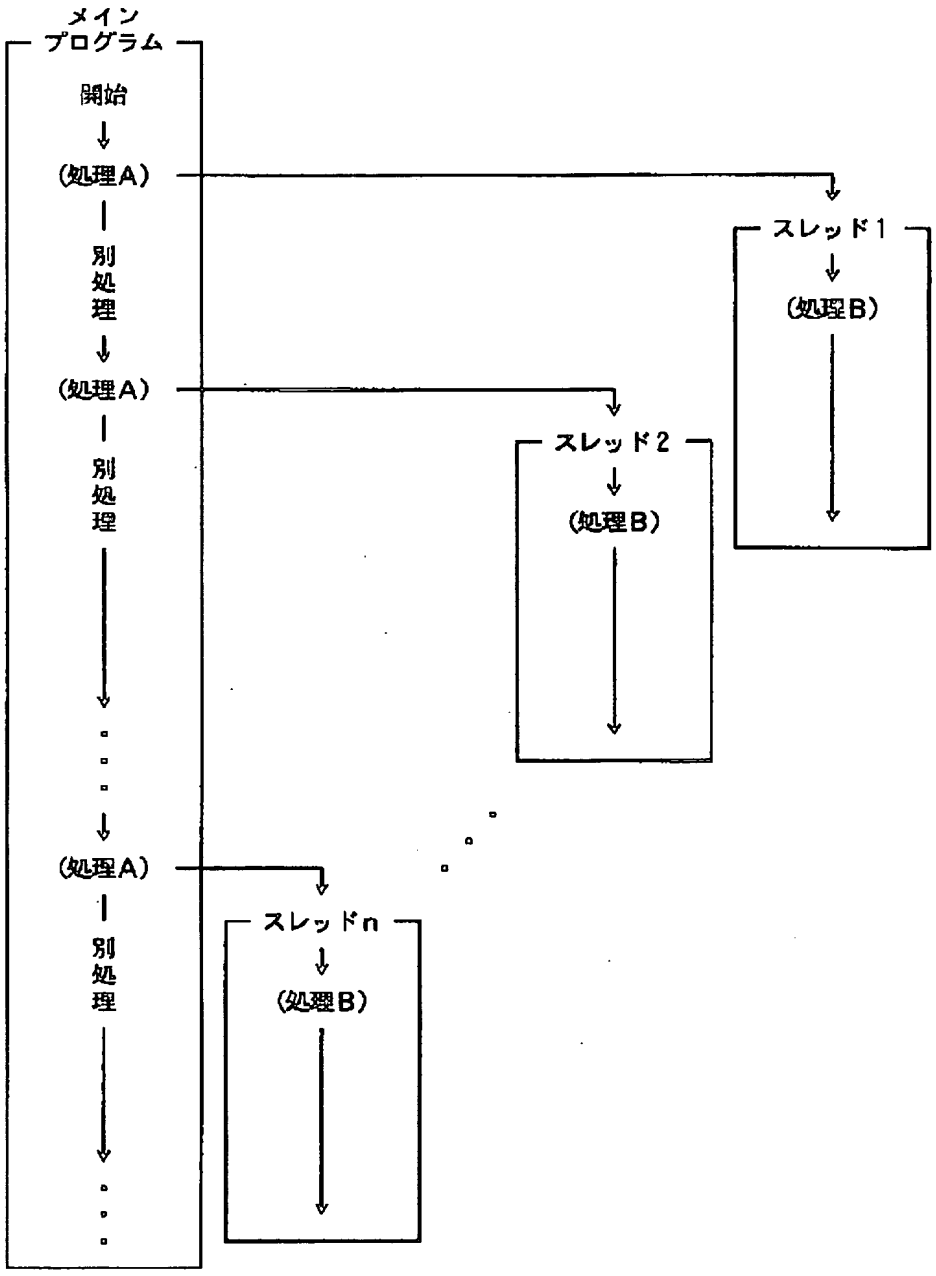
8

したフローチャート。
【図８】従来のデータ処理システムにおけるスレッド管理テーブルの説明図。
【図９】従来のデータ処理システムにおけるスレッド数の推移の説明図。
【符号の説明】
Ａスレッド番号
Ｂ　使用／未使用フラグ
Ｃ　待機フラグ
Ｄ処理データ
Ｇリサイクルスレッドカウンタ
Ｆ　共用メモリ

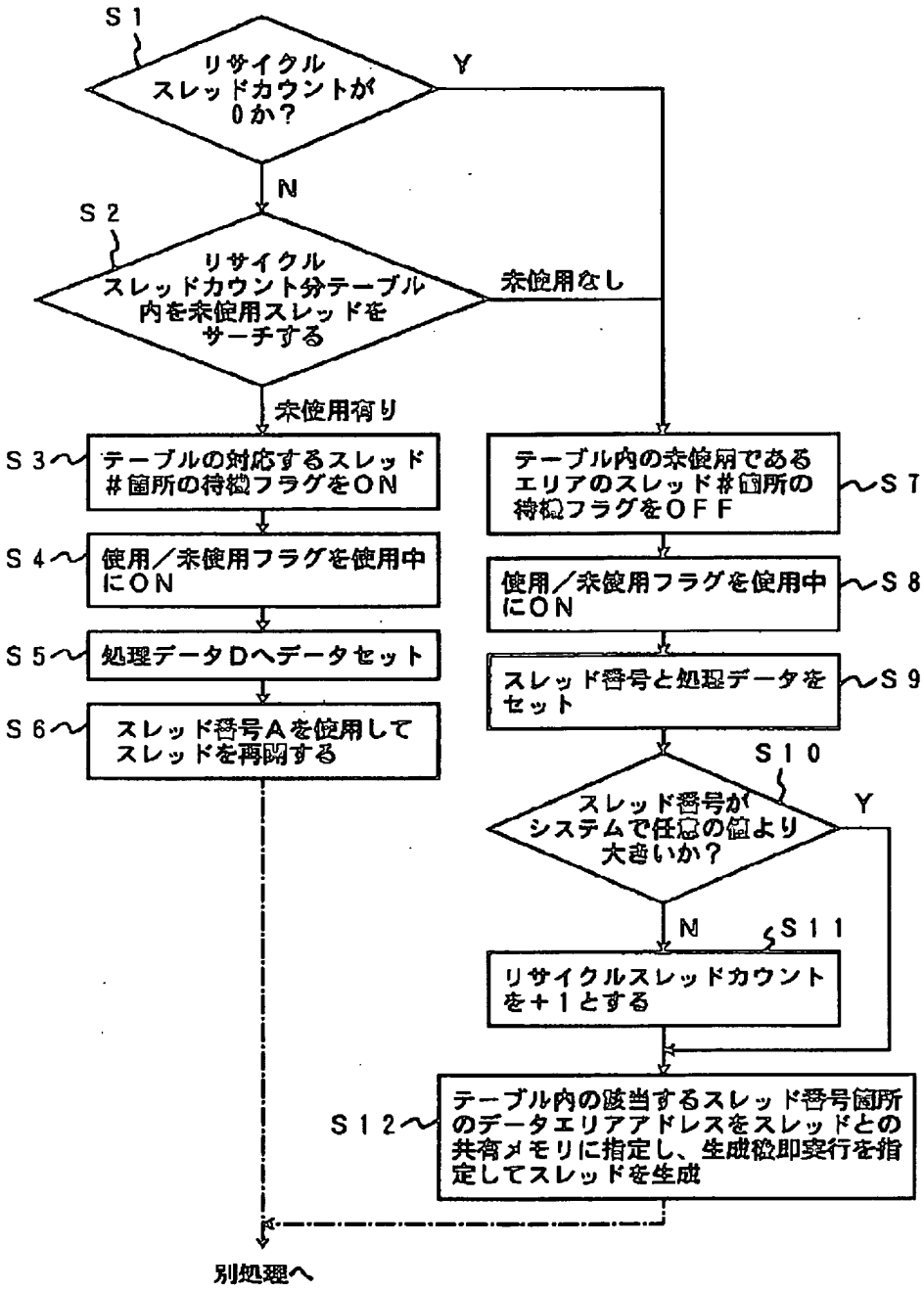
【図３】



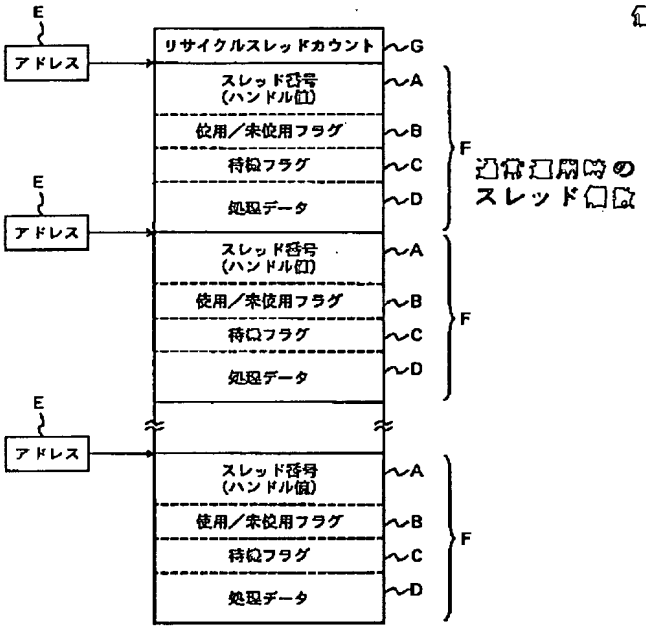
【図１】


メイン
プログラム
開始
(処理A)
別
処
理
(処理A)
別
処
理
(処理A)
別
処
理
スレッド1
(処理B)
スレッド2
(処理B)
スレッドn
(処理B)

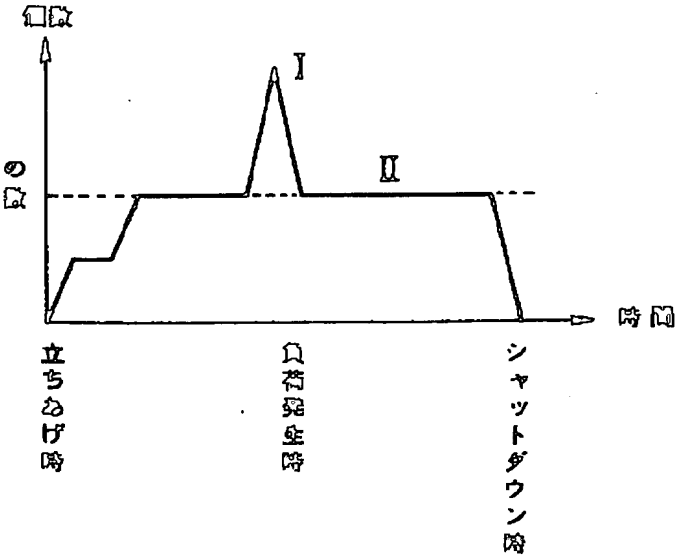
【図２】


S１
リサイクル
スレッドカウントが
0か？
Y
N
S２
リサイクル
スレッドカウント分テーブル
内を未使用スレッドを
サーチする
未使用なし
未使用有り
S３
テーブルの対応するスレッド
＃箇所の待機フラグをＯＮ
S４
使用／未使用フラグを使用中
にＯＮ
S５
処理データＤへデータセット
S６
スレッド番号Ａを使用して
スレッドを再開する
別処理へ
テーブル内の未使用である
エリアのスレッド＃箇所の
待機フラグをＯＦＦ
S７
使用／未使用フラグを使用中
にＯＮ
S８
スレッド番号と処理データを
セット
S９
S１０
スレッド番号が
システムで任意の値より
大きいか？
Y
N
S１１
リサイクルスレッドカウント
を＋１とする
S１２
テーブル内の該当するスレッド番号箇所
のデータエリアアドレスをスレッドとの
共有メモリに指定し、生成後即実行を指
定してスレッドを生成

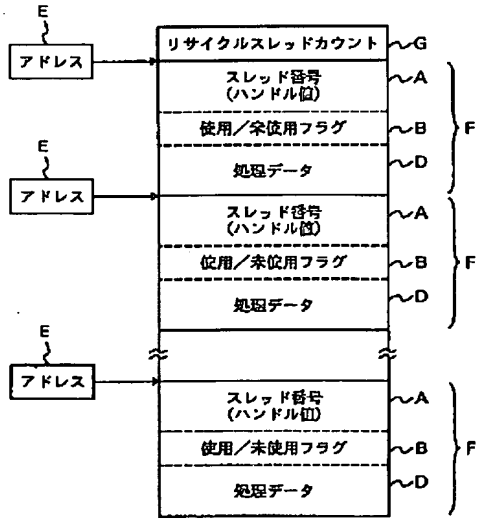
【図４】


E
アドレス
リサイクルスレッドカウント
G
スレッド番号（ハンドル値）
A
使用／未使用フラグ
B
待機フラグ
C
処理データ
D
F
スレッド構造体

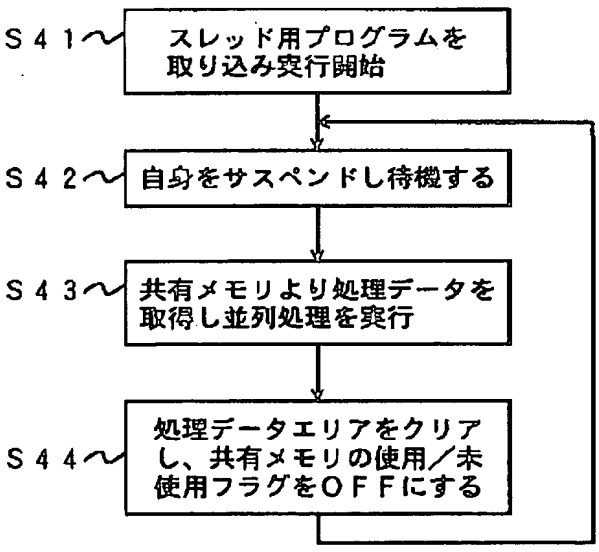
【図５】


負荷
I
II
時間
立ち上げ時
負荷発生時
シャットダウン時

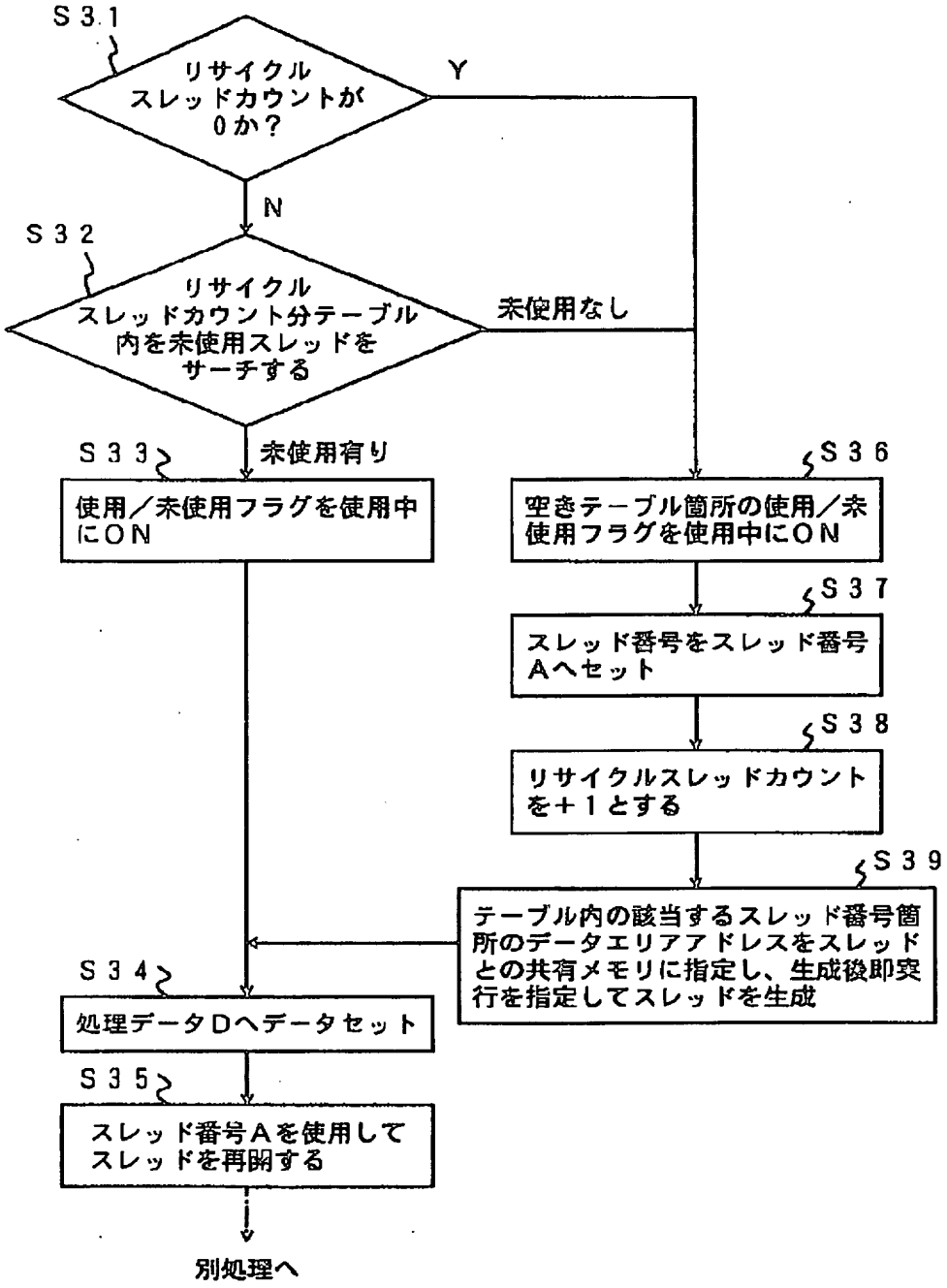
【図８】


E
アドレス
リサイクルスレッドカウント
G
スレッド番号（ハンドル値）
A
使用／未使用フラグ
B
処理データ
D
F

【図７】


S４１
スレッド用プログラムを取り込み実行開始
S４２
自身をサスペンドし待機する
S４３
共有メモリより処理データを取得し並列処理を実行
S４４
処理データエリアをクリアし、共有メモリの使用／未使用フラグをＯＦＦにする

【図６】


S31
リサイクル
スレッドカウントが
0か？
Y
N
S32
リサイクル
スレッドカウント分テーブル
内を未使用スレッドを
サーチする
未使用なし
S33
未使用有り
使用／未使用フラグを使用中
にON
S36
空きテーブル箇所の使用／未
使用フラグを使用中にON
S37
スレッド番号をスレッド番号
Aへセット
S38
リサイクルスレッドカウント
を＋1とする
S39
テーブル内の該当するスレッド番号箇
所のデータエリアアドレスをスレッド
との共有メモリに指定し、生成後即実
行を指定してスレッドを生成
S34
処理データDへデータセット
S35
スレッド番号Aを使用して
スレッドを再開する
別処理へ

【図９】